View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

Generation and Exploitation of Aggregation Abstractions for Scheduling and Resource Allocation

Theodore A. Linden
Advanced Decision Systems
1500 Plymouth St.
Mountain View, CA 94043
linden@ads.com

Michael R. Lowry
Kestrel Institute
3260 Hillview Avenue
Palo Alto, CA 94306
lowry@kestrel.edu

Abstract

Our research is investigating abstraction of computational theories for scheduling and resource allocation. These theories are represented in a variant of first order predicate calculus, parameterized multi-sorted logic, that facilitates specification of large problems. A particular problem is conceptually stated as a set of ground sentences that are consistent with a quantified theory. We are mainly investigating the automated generation of aggregation abstractions and approximations in which detailed resource allocation constraints are replaced by constraints between aggregate demand and capacity. We are also investigating the interaction of aggregation abstractions with the more thoroughly investigated abstractions of weakening operator preconditions. The purpose of the theories for aggregated demand/capacity is threefold: first, to answer queries about aggregate properties, such as gross feasibility; second, to reduce computational costs by using the solution of aggregate problems to guide the solution of detailed problems; and third, to facilitate reformulating theories to approximate problems for which there are efficient problem solving methods. We also describe novel methods for exploiting aggregation abstractions.

Motivation

Domain specific planning and scheduling systems have achieved a modicum of real world success, and current efforts are aimed at vastly increasing the size and complexity of problems which can be handled with knowledge-based technology. We believe that much of the power of domain-specific planning and scheduling systems comes from their use of specialized algorithms at different levels of abstraction. For example, a resource allocation problem can often be approximated as linearized upper and lower bounds at a high level of abstraction, and solved using linear programming methods in order to identify bottleneck resources. Domain-specific scheduling systems use many different kinds of abstraction, not just the abstraction hierarchies defined by dropping literals from operator preconditions, as is the case for ABSTRIPS and most of its progeny. In particular, for large scheduling and resource allocation problems whose computational complexity is characterized by resource contention between many separate tasks, aggregation abstractions of demand and resource capacity play a more dominant role than abstraction of operator preconditions. An example is to aggregate all transportation capacity into a single linear quantity - total cargo volume. However, the drawback of domain-specific systems is their lack of flexibility and the necessity of hand-coding the knowledge.

The objective of our research is to develop the technology for dynamically 'compiling' domain-specific scheduling systems from declarative specifications and the subgoals and constraints that arise during planning and scheduling. The goal is to achieve the efficiency of hand-coded domain-specific systems but at the same time maintain the benefits of domain independent systems which interpret declarative problem specifications. The benefits of the latter arise from their generality: because the assumptions are explicit rather than hard-coded, the system is more widely applicable, the declarative representation is more transparent and thus more trusted and more easily validated, and furthermore the representation is more easily modified as requirements evolve. The automated synthesis and selection of abstractions is a key component to enabling domain-specific systems to be compiled from declarative specifications.

The next section of this paper describes the underlying semantics we are using for abstractions, approximations, and aggregations. The subsequent section describes the techniques we are developing for generating abstractions and approximations. The final section describes new techniques for exploiting aggregation approximations and abstractions.

Semantics of Abstractions, Approximations, and Aggregations

Semantically, we define an abstraction as a (possibly partial) mapping from the models of one theory to the models of another theory. We assume that these mappings are transitive and reflexive. If a mapping is total and can be inverted, then the two theories it relates are isomorphic.

The intended semantics of a theory determine the appropriate constraints on a valid abstraction. We define the appropriate abstraction constraints through the converse mapping: implementation. For *loose* specifications, the intended semantics is any model satisfying the theory, hence a valid implementation is a mapping from the models of the implementing theory into (but not necessarily onto) the models of the loose specification. This is compatible with definitions of abstraction as theory generalization, i.e. a widening of the class of models. For this type of semantics, implementation is the same as theory refinement.

For *tight* specifications, the intended semantics is a minimal model (up to isomorphism). Minimal model semantics correspond to the operational semantics of most types of logic programming. Minimal model semantics are also useful for succinctly axiomatizing models with inductive types, the simplest being the natural numbers. Hence the appropriate constraint on abstraction is a mapping from a single concrete model to a single abstract model. A third type of specification is *parameterized*, consisting of a parameter theory, which has many interpretations, and a body which extends the parameter theory. The usual intended semantics for this type of specification is to take *all* the models of the parameter theory, and then to extend each one with additional objects, functions and relations such that this extension is minimal with respect to all possible extensions consistent with the body. This third type of semantics is best seen as a tight specification for each model of the parameter theory. This type of semantics is most useful when a general specification is given for a whole class of problems.

We have axiomatized various types of generic resources using parameterized specifications: consumable resources (such as fuel), reusable non-shareable resources (such as a landing strip), synchronized-shareable resources (such as a cargo ship), and independent-shareable resources (such as a parking lot). A particular domain theory is built up by composing instantiations of these generic parameterized resource theories with particular resources.

Syntactically, an abstraction is defined through two theories and a set of definitions for abstraction functions from the objects and operations of the concrete model(s) to the objects and operations of the abstract model(s). The abstraction functions are defined in the syntax of the union of the abstract and concrete theories.

Approximations arise from weakening or strengthening the criteria for models of a theory. In the context of our research, this weakening/strengthening is always with respect to queries or goals. For example, if the goal is to transport cargo from one country to another given a certain set of resources, then the satisfaction of a strengthened approximation guarantees the transportation feasibility of the original, while the non-satisfaction of a weakened approximation guarantees the transportation infeasibility of the original. Strengthening and weakening occur not only with respect to the truth of sentences but also with respect to any partial order, such as the total order on the reals (true/false defines a partial order on the booleans). For example, approximations can also be upper and lower bounds on resources required for transportation feasibility. Given a complex query or goal it is necessary to map strengthening/weakening of the whole into strengthening/weakening of the parts. The polarity analysis for sentences [Manna & Waldinger 86] has been extended to a polarity analysis for any type of formula ranging over any domain with a partial order [Smith 92]. Thus given a complex query with a specified direction of strengthening or weakening, the constraints on the strengthening/weakening of the functions and relations in the query can be mechanically derived.

Aggregations are mappings from collections of objects with their individual attributes to a whole representing the collection with attributes for the collective. In theory aggregations can arise as equivalent conditions for satisfaction of a goal. For example, in order for a chemical reaction to occur in a solution the individual molecules must have sufficient kinetic energy. This constraint on the attributes of individual molecules can be reformulated into an equivalent constraint on the temperature of the whole solution. In the context of the research reported in this paper aggregations are most often approximations with respect to a query or goal.

Generation of Aggregation Abstractions

We are using two techniques for automatic generation of aggregation approximations. The first is based on analysis of behavioral equivalence: given a goal, two objects are behaviorally equivalent if they can be mutually substituted for each other in the achievement of the goal. For example, for the goal of transporting a rifle division, two small cargo planes are behaviorally equivalent to one large cargo plane. However, for transporting a heavy armor division only the large cargo plane has a sufficient girth for tanks and heavy artillery. Thus for transporting heavy armor divisions two small cargo planes are not behaviorally equivalent to a single large cargo plane. This simple example illustrates that abstractions such as total lift capacity must be dependent on context in order to be useful.

The result of behavioral equivalence analysis is the definition of an equivalence relation on the objects of a domain. In an abstract theory, behaviorally equivalent objects are identified. Syntactically, the equivalence relation in the concrete theory is transformed to an equality relation in the abstract theory. A number of issues arise in ensuring that the transformation from a behavioral equivalence relation to an equality relation is semantically well defined, particularly for inductively defined types. These issues are addressed in [Lowry 1989, Lowry 1990].

When behavioral equivalence analysis is applied to a set of goals, or to a complex domain theory with many constraints, the result will be a set of behavioral equivalences. (The behavioral equivalence for the conjunction of the goals is the intersection of the individual equivalence relations.) This set can be ordered by inclusion, defining a partial order on behavioral equivalence relations.

Upper and lower bounds exist for each pair of behavioral equivalence relations; a lattice is defined by including the universal equivalence relation (all objects equivalent) and the identity equivalence relation (each object is equivalent only to itself). This lattice can be more densely ordered by inheriting ordering relations on the goals and constraints. For example, ABSTRIPS type orderings on literals in the preconditions of operators derived through various programs [Tenenberg 89; Knoblock 89, 90] can be used to order their corresponding behavioral equivalence relations.

The second technique for generation of aggregation approximations is through the use of bounding approximations: given a goal or query, and an abstract domain theory obtained through behavioral equivalence analysis, an extended polarity analysis is applied to the goal(s) with respect to the abstract domain theory. Various kinds of symbolic bounding approximations are derived by KIDS through this polarity analysis, which is currently implemented through a transitive rewriting technique on formulas called directed inference [Smith 90]. These approximations include necessary conditions, sufficient conditions, symbolic upper bounds, and symbolic lower bounds. These approximation functions are composed with the abstraction functions derived through behavioral equivalence analysis to yield the abstraction (approximation) functions that map the concrete domain onto the abstract domain.

To derive aggregation approximations, the domain theory must include generic axioms defining the relation between aggregate constraints and constraints on individuals. Most of the resources we are considering are linear: resources have time-varying capacities which at each instance cannot be exceeded by the sum of the consumers assigned to that resource. The axioms for these kinds of generic aggregation constraints, together with a particular domain theory, are transformed by the polarity analysis into definitions of possible aggregation approximations.

Like the behavioral equivalence abstractions, the aggregation approximations derived through polarity analysis can be ordered by strength. Furthermore, directed inference, because it is a transitive rewriting technique, automatically generates part of this ordering relation. The composition of the lattice of behavioral equivalence relations and the ordering on aggregation approximations again yields a lattice. We are investigating techniques for this composed lattice to be implicitly defined rather than explicitly generated.

Exploiting Resource Abstractions

For very large resource allocation and scheduling problems that are solved interactively, some form of greedy algorithm is often appropriate. In this section, we illustrate the use of resource abstraction hierarchies to enhance the opportunities for finding a linear ordering of allocation and scheduling decisions that achieves good results within the context of a greedy algorithm. The same resource abstraction hierarchies can also be used to enhance the variable ordering heuristics used with other search strategies.

The example described later in this section shows that resource abstractions can enable a successful linear ordering of resource allocation decisions where no such ordering exists without the abstractions. The resource abstractions allow allocation decisions (assignments of resources to operations) to be made in steps down the resource abstraction hierarchy; first an abstract resource is allocated to an operation, later, this decision is refined to a more specific resource. Each allocation of an abstract resource is essentially a reservation for an unspecified instance of that abstract group of resources. By making the reservation early while deferring more specific resource choices, the deferred decisions can take advantage of information that becomes available as other decisions are made.

The cost of using resource abstraction hierarchies is the additional checking needed to maintain consistency of the allocation as it is built incrementally. Each allocation decision must be checked to ensure that it does not overuse the resource that is being assigned. With abstract resources, each decision must be checked not only against the resource being assigned but also against the more abstract resources.

To formalize these concepts, assume we have a resource abstraction hierarchy in the form of a lattice (**R**, »=) where **R** is a set of resource types and »= is the **extends relation** which is a partial ordering defined on **R** meaning "is more specific than (or equal to)." In the example discussed later in this section, the specific resources are seaports and **R** consists of individual seaports and selected sets of seaports. The lattice relation on **R** is membership or subset. For example, Norfolk » tank-loading seaports » East Coast seaports. Similarly, Baltimore » mid-Atlantic seaports.

Given a set of specific resources, the power set of these resources is a candidate lattice; however, this is typically a bad candidate because it would involve exponential cost in checking resource assignments against the more abstract nodes. While a narrow lattice can be developed at design time; it is likely to be far more effective to choose abstractions that are tailored to the specific problem instance using the techniques discussed previously of behavioral equivalence analysis and aggregation approximations.

Figure 1 depicts a small portion of a simplified problem involving military crisis action deployments. A large number of force modules (such as all the equipment associated with a brigade) are to be shipped through East Coast ports. Two of the hundreds of such force modules (FM15 and FM35) are identified in the figure. The problem addressed in this example is to plan the deployment without excessive congestion at the ports (other aspects of the overall problem involve scheduling of transports and other resources).

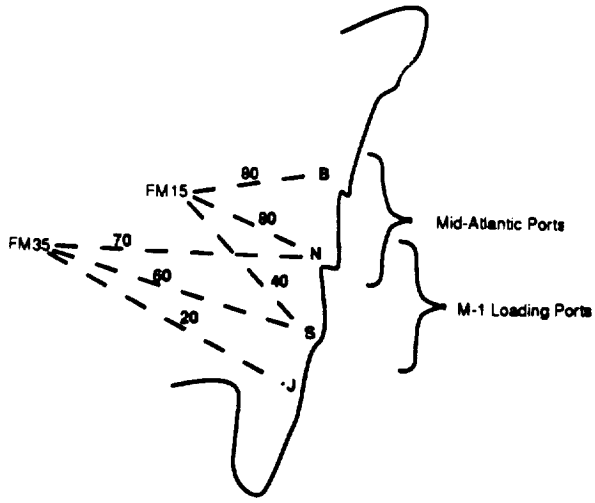

Figure 1: Problem of assigning seaports to transportation tasks

The two force modules can be shipped through any of 4 different ports, labeled B, N, S, and J. The dashed lines between the force modules and the ports that are suitable for shipment are labeled with the utility of using this route. That utility summarizes the current state of information about the cost of ground transportation to the port, the locations of available transport ships, characteristics of the ports, and other factors.

If this port selection problem could be isolated from other aspects of the overall crisis action planning problem, it could be solved by existing OR algorithms. Unfortunately, there are many complex dependencies between the port selection and the scheduling of other resources that require the port selection decisions to be interleaved with other decisions within an overall multi-user, interactive construction paradigm. Within this context, the port selection is done with a greedy algorithm, and a goal is to use the locally available information to order the allocation decisions.

This example focuses on deciding whether to choose a port for FM15 before or after choosing one for FM30. Without abstraction levels, neither ordering can be successful under all interactions with other demands on the port resources. If we use the simple greedy approach and assign to FM15 first because it can achieve a higher utility, then the arbitrary choice that has be made between B and N may prevent FM35 from obtaining its best choice. (We can't choose B knowing that FM35 prefers N because we are assuming there are many other force modules that are also competing for both B and N, and many of them may have higher priority than FM35.) On the other hand, if the assignment is made to FM35 first, it may choose port N, but that could be the cause of FM15 not getting either of its good choices. This problem of ordering these two allocation decisions (and the decisions about all other pairs of force modules) might be avoided by using statistical look-ahead techniques [Fox 89] that project the contention at the ports and may allow FM15 to choose first and make a non-arbitrary choice between B and N. Separating the decisions across abstraction level gives additional opportunities to be successful with a greedy algorithm, and appears to be especially useful in conjunction with statistical look-ahead techniques. Note that no technique can make a greedy algorithm successful all the time.

Two abstractions on the seaports are shown in Figure 1—mid-Atlantic ports and M-1 loading ports. For the abstractions to be effective, the utility function should often be more homogeneous across different instances of the abstract resource than across the entire domain of the resources. This can be a goal when creating abstractions dynamically.

The mid-Atlantic port abstraction enables an ordering of the port allocation decisions for these two force modules that will almost always be successful:

1) Reserve a mid-Atlantic port for FM15 (assuming it competes successfully with other force modules for these ports).

2) Reserve an M-1 loading port for FM35 (assuming it competes successfully with other force modules for these ports).

3) Assign N to FM35 assuming it preserves all reservations for both mid-Atlantic and M-1 loading ports and competes successfully for N with the other force modules.

4) Assign to FM15 whichever instance of a mid-Atlantic port is left over. (The reservation guarantees that some mid-Atlantic port will be left for FM15)

The reservation that FM15 has for a mid-Atlantic port allows a lower priority force module to be given precedence over the higher priority module in step three—as long as the reservation is preserved.

This four step ordering of the decisions about these two force modules often achieves a good solution without backtracking. Similar reasoning for other pairs of force modules can be used to order the other decisions made by a greedy algorithm—but there are no guarantees that a good ordering can be found for all pairs of decisions.

Bibliography

[Fox 89] Mark S. Fox et al, "Constrained Heuristic Search", Proc. IJCAI-89, Morgan Kaufmann Publ.

[Knoblock 89] Craig A. Knoblock, "A Theory of Abstraction for Hierarchical Planning", *Change of Representation and Inductive Bias*, ed. D.P. Benjamin, Kluwer 1989

[Knoblock 90] Craig A. Knoblock, "Learning Problem-Specific Abstraction Hierarchies", Proc. of Workshop on Change of Representation and Problem Reformulation; Menlo Park, CA March 1990

[Linden 89] Theodore A. Linden, "Planning by Transformational Synthesis," *IEEE Expert*, 4,2 Summer, 1989, pp. 46-55.

[Linden 90] Theodore A. Linden, "Transformational Synthesis: A Paradigm for Building Large-Scale Planning Applications," *Planning Systems for Autonomous Mobile Robots*, ed. D. P. Miller and D. J. Atkinson, Prentice Hall, 1989.

[Lowry 89] Michael R. Lowry, "STRATA: Problem Reformulation and Abstract Data Types", in Change of Representation and Inductive Bias, Edited by Paul Benjamin, Kluwer Academic Publishers 1989

[Lowry 89] Michael R. Lowry, "Algorithm Synthesis through Problem Reformulation", PhD Thesis, Stanford University, 1989

[Lowry 90] Michael R. Lowry, "Abstracting Domains with Hidden State", in Proc. of Workshop on Automatically Generating Abstractions and Approximations, AAAI-90

[Manna & Waldinger 86] Manna, Z. and Waldinger, R. Special Relations in automated deduction. *Jorunal of tthe ACM 33, 1 pp.* 1-59.

[Smith-Lowry-89] Smith, D.R. and Lowry, M.R., Algorithm Theories and Design Tactics, in Proceedings of the International Conference on Mathematics of Program Construction, LNCS-375, Springer-Verlag, Berlin, June 1989, 379-398 (to appear in Science of Computer Programming, 1990).

[Smith-90] Smith, D.R., KIDS: A Semi-Automated Program Development System, in IEEE Transactions on Software Engineering special issue on Formal Methods, September 1990.

[Smith-92] Smith, D.R., Connecting Specification Morphisms, in Proceedings of the International Conference on Mathematical Theory of Computation.

[Tenenberg 89] Josh Tenenberg, "Abstracting First-Order Theories", *Change of Representation and Inductive Bias*, ed. D.P. Benjamin, Kluwer 1989